

3-23-00
03/21/00
JCT88 U.S. PTO

3-23-00

A

PATENT APPLICATION
Attorney's Do. No. 1467-13

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

EXPRESS MAIL

MAILING LABEL NO. EL433178381US
DATE OF DEPOSIT: MARCH 21, 2000

I HEREBY CERTIFY THAT THIS PAPER AND ENCLOSURES AND/OR FEE ARE BEING DEPOSITED WITH THE UNITED STATES POSTAL SERVICE "EXPRESS MAIL POST OFFICE TO ADDRESSEE" SERVICE UNDER 37 CFR 1.10 ON THE DATE INDICATED ABOVE AND IS ADDRESSED TO: BOX PATENT APPLICATION, ASSISTANT COMMISSIONER FOR PATENTS, WASHINGTON D.C. 20231.

Amanda Hale-Wisener
(SENDER'S PRINTED NAME)

Amanda Hale-Wisener
(SIGNATURE)

PTO
09/531350
03/21/00

Box Patent Application
Assistant Commissioner for Patents
Washington, D.C. 20231

Enclosed for filing is a patent application under 37 CFR 1.53(b) of:

Inventor [or Application Identifier]: Steven S. Greenberg
For: SCHEDULING NON-INTEGRAL SIMULATION TIME FOR MIXED-SIGNAL SIMULATION

[If continuing application] This application is a ☐ continuation, ☐ divisional, ☐ continuation-in-part of prior application Serial No. _____, filed _____.

Enclosures:

- ☒ Specification (pages 1-8); claims (pages 9-11); abstract (page 12)
- ☒ 12 sheet(s) of drawings
- ☒ Declaration or Combined Declaration and Power of Attorney
 - ☒ Newly executed (original or copy)
 - ☐ Copy from a prior application (37 CFR 1.63(d))
 - ☐ Incorporation by Reference--The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.
 - ☐ Deletion of Inventors (signed statement attached deleting inventor(s) named in the prior application (37 CFR 1.63(d)(2) and 1.33(b))
- ☒ Verified Statement Claiming Small Entity Status
- ☒ Power of Attorney
- ☒ Assignment with cover sheet

- ☐ Certified copy of priority document:
☒ Information Disclosure Statement with Form PTO 1449
☒ Copies of references listed on attached Form PTO-1449
☐ Preliminary Amendment
☐ Change of Address
☒ Return Postcard

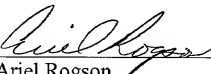
CLAIMS AS FILED				
For	Number Filed	Number Extra	Rate	Basic Fee
Total Claims	18-20	0	x \$ 9 =	\$345.00 0.00
Independent Claims	4-3	1	x \$ 39 =	39.00
Multiple Dependent Claim Fee			x \$130 =	0.00
TOTAL FILING FEE				\$384.00

- ☐ Cancel in this divisional application original claims _____ of the prior application Serial No. _____ before calculating the filing fee. (At least one original independent claim must be retained for filing purposes.)
☒ A check in the amount of \$424.00 to cover ☒ filing fee and ☒ assignment recordal fee (\$40) is enclosed.
☒ Any deficiency or overpayment should be charged or credited to deposit account number 13-1703. A duplicate copy of this sheet is enclosed.

Customer No. 20575

Respectfully submitted,

MARGER JOHNSON & McCOLLOM, P.C.


 Ariel Rogson
 Reg. No. 43,054

MARGER JOHNSON & McCOLLOM, P.C.
 1030 S.W. Morrison Street
 Portland, Oregon 97205
 (503) 222-3613

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant or Patentee: Steven S. Greenberg

Serial No.:

Filed:

For SCHEDULING NON-INTEGRAL SIMULATION TIME FOR MIXED-SIGNAL SIMULATION

DECLARATION CLAIMING SMALL ENTITY
STATUS (37 CFR 1.9(f) and 1.27 (e)) --SMALL BUSINESS CONCERN

I hereby state that I am

- ☐ the owner of the small business concern identified below:
☒ an official of the small business concern empowered to act on behalf of the concern identified below:

Name of Concern: Analogy, Inc., an Oregon corporation
Address of Concern: 9205 S.W. Gemini Drive, Beaverton, Oregon 97008

I hereby state that the above-identified small business concern qualifies as a small business concern as defined in 13 CFR 121.3-18, and reproduced in 37 CFR 1.9(d), for purposes of paying reduced fees under Section 41(a) and (b) of Title 35, United States Code, in that the number of employees of the concern, including those of its affiliates, does not exceed 500 persons. For purposes of this statement, (1) the number of employees of the business concern is the average over the previous fiscal year of the concern of the persons employed on a full-time, part-time or temporary basis during each of the pay periods of the fiscal year, and (2) concerns are affiliates of each other when either, directly or indirectly, one concern controls or has the power to control the other, or a third party or parties controls or has the power to control both.

I hereby state that rights under contract or law have been conveyed to and remain with the small business concern identified above with regard to the invention, entitled SCHEDULING NON-INTEGRAL SIMULATION TIME FOR MIXED-SIGNAL SIMULATION

by inventor(s) _____
described in the ☒ specification filed herewith
☐ application serial no. _____, filed _____,
☐ patent no. _____, issued _____

If the rights held by the above-identified small business concern are not exclusive, each individual, concern organization having rights to the invention is listed below * and no rights to the invention are held by any person, other than the inventor, who could not qualify as a small business concern under 37 CFR 1.9(d) or by any concern which would not qualify as a small business concern under 37 CFR 1.9(d) or a nonprofit organization under 37 CFR 1.9(e)

*NOTE: Separate statements are required from each named person, concern or organization having rights to the invention averring to their status as small entities. (37 CFR 1.27)

FULL NAME: _____

ADDRESS: _____

☐ individual ☒ small business concern ☐ nonprofit organization

I acknowledge the duty to file, in this application or patent, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small entity is no longer appropriate. (37 CFR 1.28(b))

NAME OF PERSON SIGNING Howard Ko

TITLE OF PERSON OTHER THAN OWNER Vice President of Engineering

ADDRESS OF PERSON SIGNING Analogy, Inc., 9205 S.W. Gemini Drive, Beaverton, Oregon 97008

Signature



Date

3/17/00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Steven S. Greenberg

Serial:

Filed:

For: SCHEDULING NON-INTEGRAL SIMULATION TIME FOR MIXED-SIGNAL SIMULATION

**DECLARATION CLAIMING SMALL ENTITY
STATUS (37 CFR 1.9(f) and 1.27 (b)) INDEPENDENT INVENTOR**

As a below-named inventor, I hereby state that I qualify as an independent inventor as defined in 37 CFR 1.9(c) for purposes of paying reduced fees under Section 41(a) and (b) of Title 35, United States Code, to the Patent and Trademark Office with regard to the invention entitled SCHEDULING NON-INTEGRAL SIMULATION TIME FOR MIXED-SIGNAL SIMULATION described in the

☒ specification filed herewith

☐ application serial no. _____, filed _____,

☐ patent no. _____, issued _____

I have not assigned, granted, conveyed or licensed and am under no obligation under contract or law to assign, grant, convey or license, any rights in the invention to any person who could not be classified as an independent inventor under 37 CFR 1.9(c) if that person had made the invention, or to any concern which would not qualify as a small business concern under 37 CFR 1.9(d) or a nonprofit organization under 37 CFR 1.9(e).

Each person, concern or organization to which I have assigned, granted, conveyed, or licensed or am under an obligation under contract or law to assign, grant, convey, or license any rights in the invention is listed below:

☐ no such person, concern, or organization

☒ persons, concerns or organizations listed below*

*NOTE: Separate statements are required from each named person, concern or organization having rights to the invention averring to their status as small entities. (37 CFR 1.27)

FULL NAME: Analogy, Inc., an Oregon corporation

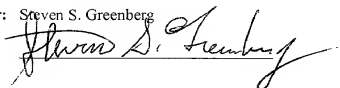
ADDRESS: 9205 S.W. Gemini Drive, Beaverton, Oregon 97008

☐ individual ☒ small business concern ☐ nonprofit organization

I acknowledge the duty to file, in this application or patent, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small entity is no longer appropriate. (37 CFR 1.28(b))

Full name of sole or first inventor: Steven S. Greenberg

Inventor's signature:

 3/17/2000
(Date)

SCHEDULING NON-INTEGRAL SIMULATION TIME FOR MIXED-SIGNAL SIMULATION

FIELD OF THE INVENTION

The present invention relates to the field of computer simulation of analog and mixed signal digital-analog physical circuits and systems, and more particularly to the scheduling of non-integral times for simulation events.

BACKGROUND OF THE INVENTION

In simulating digital systems to verify both their functional behavior and their timing behavior, speed of simulation is one of the most important factors. One of the key ways of speeding up simulation is for the simulation to selectively trace only the activity in the model. This selective trace occurs in two dimensions. One dimension is the model. At any given simulation time, the simulator expends no effort on the parts of the model that are not changing. The other dimension is time. Ideally, the simulator only expends effort at the simulation times at which activity occurs.

To achieve these efficiencies, activity is broken down into events that occur at specific times. As the model creates these events, it is crucial that the simulator efficiently store these events for processing at the specified simulation times and efficiently retrieve them when the simulator progresses to that simulation time. Typically, data structures and algorithms that are an implementation of a time wheel are used to accomplish this. One of the features of the time wheel is its ability to give the simulator, in one operation, the entire list of all events at the next simulation time at which an event occurs.

Each slot in the time wheel represents one unit of minimum resolvable time. A modulo operation on the simulation time is used to find the proper slot in which to place an event. Simulation time is advanced by finding the next slot in the time wheel that has a non-empty event list. The order of complexity of the modulo operation is a constant in computer time. The cost of testing each empty slot is significant enough that it is important to minimize the number of empty slots. Minimization is done by choosing a minimum resolvable time that is small enough to get the desired accuracy in time, but large enough to minimize the number of empty slots. From the simulator's point of view, it is most

convenient if this minimum resolvable time is known at the beginning of simulation. Otherwise, provision must be made for re-organizing the time wheel as simulation progresses. This re-organization is very expensive.

Another difficulty is deciding the size of the time wheel. It must be large enough to cover the largest delay in the model. This assures that two different times do not have to occupy the same slot in the time wheel at the same time. This can limit the size of delay that can be modeled in a given amount of computer memory. The other options are to manage different times in the same slot or to manage a separate list of events that are far in the future. Either of these last two options introduces complexity and inefficiency into the simulator.

Simulators can operate efficiently if the minimum resolvable time can be discerned at model compilation time. There is a much less efficient mode of simulation if the minimum resolvable time can vary during the simulation.

During the analog parts of the simulation, the minimum resolvable time is determined by the simulator as the time increment required to solve the algebraic-differential equations of the model accurately. At different times during simulation, the minimum resolvable time can vary by many orders of magnitude. This minimum resolvable time can sometimes act as a delay that needs to be scheduled. At the interface from analog to digital simulation, there are threshold crossings that need to be scheduled. The times of these threshold crossings are at analog time resolution.

FIG. 9 shows the sequence of steps involved in performing a simulation. At step 905, the design description is assembled. The design description includes any kind of input that describes the design to be simulated. It can include models describing physical devices and a netlist (a list of model instances and their interconnections and parameters). Models and the netlist can be expressed as text or in compiled form. At step 910, the simulatable model is assembled. This takes the design description and converts it into a collection of data structures and executable code that can be used by the core simulator to analyze the performance of the model representing the design to be simulated. At step 915, the core simulator simulates the model. The core simulator computes the performance of the mixed analog/digital model in the domains of time and frequency. Finally, at step 920, the post-processor allows the user to inspect the simulation results, for example in a graphical viewer, and to process the results according to selected criteria.

FIGs. 4A and 4B show the simulation process for simulating an analog and mixed signal digital-analog physical circuit or system. The simulation process shows where events are scheduled internal to the system. The simulation process of FIGs. 4A and 4B is akin to

the simulation process specified in IEEE Standard VHDL Analog and Mixed-Signal Extensions, IEEE Std 1076.1-1999, § 12.6.4, and is similar to the simulation performed by SABER.

In FIGs. 4A and 4B, at step 405, the simulator checks to see if there are more events to simulate. If there are scheduled times remaining, then there are more events to simulate. At step 410, assuming events remain to be simulated, scheduled analog events are handled. There are three kinds of analog events: arriving at the left boundary of a discontinuity, where the model to be simulated has executed a break; arrival at the right boundary of a discontinuity; and arrival at the time of the projected analog solution. At step 415, signals are updated. Updating signals includes propagating value changes to the signals. At step 420, processes are resumed. A *process* is a set of procedural steps in the hardware description language provided by the user. Processes can handle user signal changes, and can assign values to a signal. Assigning a value to a signal causes an event to be scheduled. At step 425, the timeslot is checked to see if it has ended. Because events can be scheduled with a zero delay (in other words, events can be added to be simulated immediately), the timeslot must be checked before the simulator can conclude that all scheduled events have been simulated. At step 430, assuming the end of the timeslot has been reached, postponed processes are resumed. A *postponed process* is a special process set to execute as a time change occurs. At step 435, the simulator checks to see if an analog projection is required. An analog projection is required if the previous analog projection only projected to the current scheduled time. If an analog projection is required, at step 440 the next analog solution is projected. Finally, at step 445, the time of the next analog/digital events is retrieved.

When there is a discernible minimum resolvable time, the simulation time can be represented as an integer. But when the delay between scheduled simulation times can be both very small and very large (i.e., the minimum resolvable time can vary), the time must be represented as a real number (non-integer). This, too, affects the complexity of the time wheel.

Accordingly, a need exists for a way to control the timing of event simulation that is not dependent on a fixed minimum resolvable time, and that allows for effectively unbounded growth in the number of scheduled times for simulation events.

SUMMARY OF THE INVENTION

A simulator models analog and mixed signal digital-analog physical circuits and systems in a digital computer. According to the invention, the simulator assigns scheduled times to the events. Using a non-order preserving hash function, the events are placed in buckets in a hash table. Events that hash to the same bucket but are at different times are placed within different lists in the bucket. The scheduled times for the events in each bucket are associated with each bucket by the hash function. The scheduled times for the events are organized in a heap. The simulator removes the earliest scheduled time from the heap, locates the associated bucket, finds the appropriate list within the bucket, and performs the events in the event list. The remaining scheduled times are re-organized into a new heap, and the process repeats until the heap is empty.

The foregoing and other objects, features and advantages of the invention will become more readily apparent from the following detailed description of a preferred embodiment, which proceeds with reference to the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows a computer system implementing the method and apparatus of this invention.

FIGs. 2A and 2B show sample heaps that can be used as the heap of FIG. 1.

FIGs. 3A and 3B shows a sample hash table that can be used as the hash table of FIG.

1.

FIGs. 4A and 4B show in greater detail the prior art simulation process of FIG. 9.

FIG. 5 shows how events are placed in and removed from the heap of FIG. 1.

FIG. 6 shows how events can be added to the heap of FIG. 1.

FIG. 7 shows how events can be de-scheduled from the heap of FIG. 1.

FIG. 8 shows how events are processed when their scheduled time is reached in the simulator of FIG. 1.

FIG. 9 shows the sequence of steps involved in performing a simulation.

DETAILED DESCRIPTION

The time wheel used previously is a hash table that preserves order and has a fixed interval between times in the wheel (i.e., the places in the hash table). The time values are stored in sorted order by using a modulo hashing function. The price paid for that sorted order is the need to process empty slots to get to the next time slot with events.

According to the invention, a non-order preserving hash function is used. The problem of having to determine a minimum resolvable time for managing the empty slot problem is thereby eliminated. Long delays also will not require special processing. The hash function places events into buckets according to their scheduled times (the buckets thereby forming the hash table), but the buckets themselves have no assigned order.

A heap priority queue enables the tracking of times in the scheduler. The heap priority queue allows easy locating of the next event time. The heap priority queue restores the ordering capability lost by going to a non-order preserving hash table.

A hash table bucket contains a list of event lists. Each event list is labeled with the simulation time of the events in the list. When there are collisions of different simulation times hashing to the same bucket, the mechanism of lists labeled with the scheduled time keeps it sorted out.

The use of a non-order preserving hash table and a heap priority queue creates a uniform scheme whose efficiency does not depend on the relative sizes of the minimum and maximum delays. The amount of memory used by the scheme depends on the number of events and unique times. Nor does the amount of memory used depend on the sizes of the delays.

FIG. 1 shows a computer system 105 using the method and apparatus of this invention. The computer system 105 includes a computer 110, a monitor 115, a keyboard 120, and a mouse 125. Other components may be part of the computer system 105, even though not shown in FIG. 1. For example, computer system 105 can include other input/output devices, such as a plotting device. Computer system 105 also includes simulator 130. Simulator 130 performs a simulation of analog and mixed signal digital-analog physical circuits and systems.

In the preferred embodiment, the scheduled times for the events to be simulated are stored in a heap. Computer system 105 includes heap 135 for storing the scheduled times in a heap priority queue and hash table 138 for storing event lists of events to be simulated at given scheduled times. Buckets 140 are part of hash table 138. Buckets 140 store the event lists of the actual events to be simulated. This allows heap 135 to store only the scheduled times for the events to be simulated, simplifying heap management.

FIG. 2A shows a sample heap 205. Heap 205 includes, among other nodes, nodes 210, 215, 220, 225, 230, and 235. A heap is a data structure organized as a binary tree. Nodes in the binary tree are said to have a parent-child relationship. That is, a node in heap 205 that is one level higher than another node is called the *parent* of the other node, and the

other node is called a *child* of the parent node. In a heap, each parent node is smaller than its two children (this is known as the *heap property*). For example, node 210 stores the value 2, which is less than the values of nodes 215 (5) and 220 (8), node 215 (5) has a smaller value than nodes 225 (6) and 230 (16), etc. Note that, aside from the requirement that each node in heap 205 be smaller than either of its children nodes, no order is imposed on the heap. So, although node 225, storing the value 6, is in the third row of heap 205, even though node 220, with the value 8, is in row 2 of heap 205. Similarly, node 235 has a smaller value than node 230, but is lower in the heap *on the other side of the heap*. A heap, such as heap 205, can be called *partially ordered*.

In the preferred embodiment, heap 205 is *left-filled*. That is, children nodes are added to the left-most parent of the last complete row in the heap that does not have two children. Further, if the parent to which the child is to be added has no children, the new child is added as the left child of the parent node. The requirement that heap 205 be left-filled simplifies implementation details, but is not required. Algorithms for constructing a heap from a set of values, for inserting new values into the heap, and for removing values from the heap are known in the art and will not be repeated here.

FIG. 2B shows how heap 205 from FIG. 2A can look after the earliest scheduled time is removed from the heap. In heap 255, node 235 has been re-positioned within the heap as a child of node 260. Similarly, nodes 215, 225, and 260 have been re-positioned. Note that heap 255 still satisfies the heap property.

FIG. 3A shows a hash table that can accompany heap 205 of FIG. 2A. In FIG. 3A, a hash function 305 maps the times from the heap to buckets. Note that the arrows used to show the mapping of the hash function are a visual convenience: in the preferred embodiment, the hash function takes a time and returns an index of the bucket in the hash table. In FIG. 3A, times 2 (310) and 9 (315) map to bucket 1 (320), and times 5 (325) and 6 (327) map to bucket 2 (330). Bucket 1 (320) has two event lists: one for events occurring at time 9 (320A), and one for events occurring at time 2 (320B). Bucket 2 (330) has two event lists: one for events occurring at time 5 (330A), and one for events occurring at time 6 (330B). Observe that multiple times can map to a single bucket; the scheduled times for the events in the event lists in the bucket identify which events are to occur at which times.

In FIG. 3A, event list 330A shows two events in the list: event 1 (332A) and event 2 (332B). Event list 330B has no events in its event list, represented by null pointer 335. This can occur if there were events originally scheduled for time 6 (332), but these events were de-

scheduled. In the preferred embodiment, the event list is a doubly linked list, which can be traversed in either direction.

FIG. 3B shows the hash table of FIG. 3A after a new event has been scheduled. In FIG. 3B, a new event has been scheduled for time 5 (325). Hash function 305 maps time 5 (325) to bucket 2 (330). Since an event list already exists for time 5 (325) in bucket 2 (330), a new event 332C is added to event list 330B.

In the preferred embodiment, events know their location within the event list. De-scheduling an event requires only changing the pointers of events before and after the de-scheduled event in the event list. FIG. 7 shows this process. At step 705, the pointers to the de-scheduled event from the prior and next event in the event list are changed, de-scheduling the event. (Note that, even if the event list for the de-scheduled event's scheduled time is empty, in the preferred embodiment the scheduled time is **not** removed from the heap.)

However, in an alternative embodiment, events can be de-scheduled from event lists by using hash function 305 to locate the bucket containing the event list for the time of the event to be de-scheduled, accessing the appropriate event list within the bucket, and removing the event.

If no event list exists for the time for which an event is being added, the scheduled time can be inserted into the heap. How this new scheduled time is inserted into the heap is discussed with reference to FIG. 6 below.

The invention can work with the simulator shown in FIGs. 4A and 4B. At step 405, whether any events remain to be simulated is determined by checking the heap. If the heap still holds any scheduled times, events remain to be simulated. New events can be scheduled at steps 420, 430, and 440. However, a person skilled in the art will recognize that any system requiring event scheduling, and not simulation, can benefit from the invention. For example, callbacks (the execution of software routines at pre-determined scheduled times) can be organized using the invention.

FIG. 5 shows how the heap and buckets are managed outside the context of simulation. At step 505, an event is assigned a scheduled time. At step 508, the bucket storing the event list into which the event is to be added is located. The hash function uses the scheduled time of the event to locate the appropriate bucket. At step 510, the event is added to the event list in the appropriate bucket. At step 515, the event's scheduled time is inserted into the heap, if needed (the event's scheduled time may have been added to the heap earlier). Inserting the event's scheduled time into the heap includes ensuring that, after the event's scheduled time is inserted, the heap retains the heap property. As shown by dashed line 518, steps 505, 508, 510, and 515 can be repeated for as many events as necessary. Once

the heap has been built, at step 520, the earliest scheduled time is removed from the heap. Removing the earliest scheduled time includes ensuring that the heap retains the heap property after the earliest scheduled time is removed. This is shown at step 522. At step 525, the bucket containing the events scheduled for the earliest time is located. Finally, at step 530, the events scheduled for the earliest scheduled time are simulated. As shown by dashed line 535, steps 520, 522, 525, and 530 can be repeated until the heap is emptied.

FIG. 6 shows the process used to add new scheduled events. FIG. 6 is used when new events need to be scheduled after the process of emptying the heap has begun. Effectively, FIG. 6 is an interrupt of the loop of steps 520, 522, 525, and 530 of FIG. 5. FIG. 6 (as well as FIGs. 7 and 8) is not specifically directed toward scheduling new simulation events, but as discussed above, the process of FIG. 6 can be applied to simulation events. At step 605, the new event is assigned a scheduled time. At step 608, the bucket storing the event list into which the event is to be added is located. At step 610, the new event is added to the event list in the located bucket. At decision point 615, the process checks to see if the new event needs to add a new scheduled time to the heap. If a new scheduled time is to be added to the heap, then at step 620 the newly scheduled time is added to the heap, and at step 625 the scheduled time are re-organized into a new heap. If more events are to be scheduled, the process can be repeated, as shown by dashed line 630.

FIG. 8 shows how events are processed when their scheduled time is reached in the simulator of FIG. 1. At step 805, a scheduled time is removed from the heap. At step 808, the remaining scheduled times in the heap are re-organized to restore the heap property. At step 810, the bucket associated with the selected scheduled time is located. Then, at decision point 815, the bucket is checked to see if there is an event list for the scheduled time. If an event list for the scheduled time exists, then at step 820 the events in the event list are performed.

Having described and illustrated the principles of the invention in a preferred embodiment thereof, it should be apparent that the invention can be modified in arrangement and detail without departing from such principles. I claim all modifications and variations coming within the spirit and scope of the following claims.

We claim:

1. A method for scheduling events occurring at non-integral times in a simulation model for modeling analog and mixed signal digital-analog physical circuits and systems in a digital computer, the method comprising:

assigning scheduled times to the events;
using a hash function based on the scheduled times of the events, storing the events in buckets, each bucket containing at least one event;
associating the scheduled times assigned to the events in the buckets with the buckets;
organizing the scheduled times into a heap;
removing an earliest scheduled time from the heap;
simulating the events in the bucket associated with the earliest scheduled time;
re-organizing the remaining scheduled times into a new heap; and
repeating the steps of removing a scheduled time, simulating the events, and re-organizing the remaining scheduled times until the heap is empty.

2. A method according to claim 1, the method further comprising:

beginning the mixed-signal simulation;
determining the events from the mixed-signal simulation; and
determining the scheduled times for when the events are to occur.

3. A method according to claim 1, wherein simulating the events includes:

determining new events of the mixed-signal simulation;
determining the scheduled times for when the events are to occur; and
placing the new determined events into buckets using the hash function based on the scheduled times of the events.

4. A method according to claim 3, wherein placing the new determined events includes:

placing a first new determined event into a new bucket;
associating the scheduled time assigned to the first new determined event with the new bucket;
adding the new scheduled time associated with the new bucket to the heap; and
re-organizing the scheduled times into a new heap.

5. A method for scheduling events with associated scheduled times occurring at non-integral time intervals, the method comprising:

storing events in buckets according to their scheduled times; and
organizing the scheduled times into a structure, wherein the structure is constructed and arranged to allow easy location of an earliest scheduled time.

6. A method according to claim 5, wherein storing events includes using a non-order preserving hash table to place the events into the buckets according to the scheduled times associated with the events.

7. A method according to claim 5, wherein each bucket is assigned a specific scheduled time and stores all events assigned that scheduled time.

8. A method according to claim 5, wherein the structure containing the scheduled times is organized as a heap.

9. A method according to claim 8, wherein the method further comprises:
removing a scheduled time from the heap; and
re-organizing the remaining scheduled times into a new heap.

10. A method according to claim 9, wherein removing a scheduled time includes performing the events in the bucket associated with the removed scheduled time.

11. A method according to claim 10, wherein performing the events includes checking to see if the bucket associated with the removed schedule time is empty.

12. A method according to claim 5, wherein the method further comprises:
adding a new scheduled time to the heap; and
re-organizing the scheduled times into a new heap.

13. A method according to claim 5, wherein the method further comprises removing a de-scheduled event from a second bucket.

14. A computer-readable medium containing a program for scheduling events with associated scheduled times occurring at non-integral time intervals, the program comprising:

storage software to store events in buckets according to their scheduled times; and
organization software to organize the scheduled times into a structure, wherein the structure is constructed and arranged to allow easy location of an earliest scheduled time.

15. A computer-readable medium containing a program according to claim 14, the storage software including hash software to use a non-order preserving hash table to place the events into the buckets according to the scheduled times associated with the events.

16. A system for scheduling events with associated scheduled times occurring at non-integral time intervals, the system comprising:

a computer;

a plurality of buckets stored in the computer, each bucket storing events to occur at a scheduled time; and

a structure organizing the scheduled times, wherein the structure is constructed and arranged to allow easy location of an earliest scheduled time.

17. A system according to claim 16, wherein the structure containing the scheduled times is organized as a heap.

18. A system according to claim 16, wherein the computer includes a mixed-signal simulator for generating events and assigning times to generated events.

SCHEDULING NON-INTEGRAL SIMULATION TIME FOR MIXED-SIGNAL SIMULATION

ABSTRACT OF THE INVENTION

In the simulation of an analog and mixed-signal analog-digital physical circuit, events are assigned scheduled times. The events are stored in buckets in a hash table, with the scheduled times of the events in each bucket associated with the bucket. The scheduled times are organized into a heap, with the earliest scheduled time at the root of the heap. The earliest scheduled time is removed from the heap, and the events in the associated bucket are performed. Performing the scheduled events can cause new events to be scheduled, and existing events to be de-scheduled. When all the events in the bucket associated with the earliest scheduled time are simulated, the remaining scheduled times are re-organized into a new heap, and the steps of removing the earliest scheduled time, performing the scheduled events, and re-organizing the remaining scheduled times are repeated.

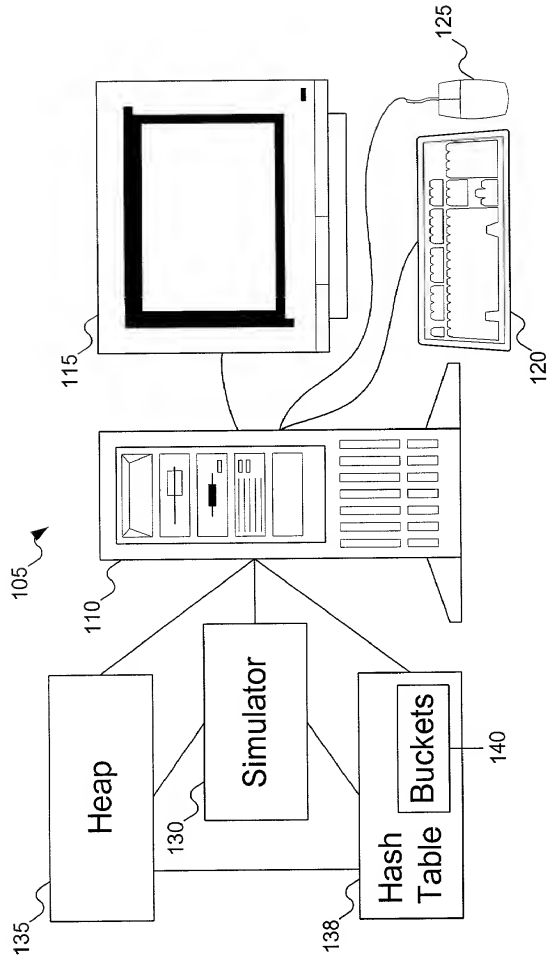


FIG. 1

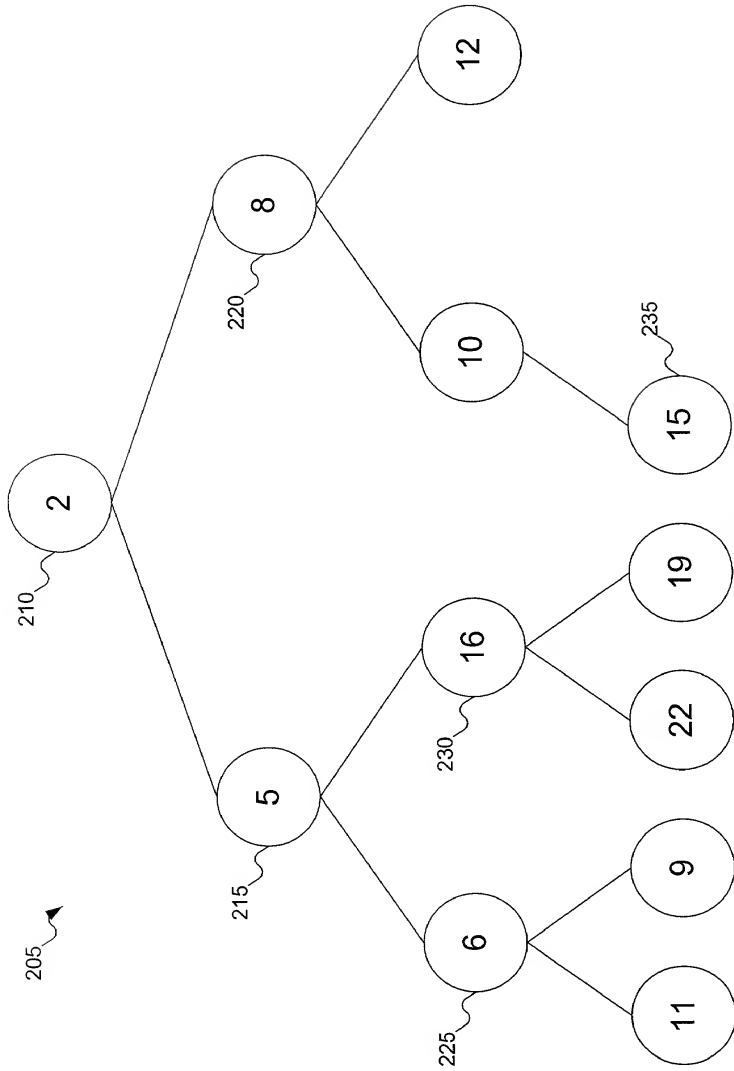


FIG. 2A

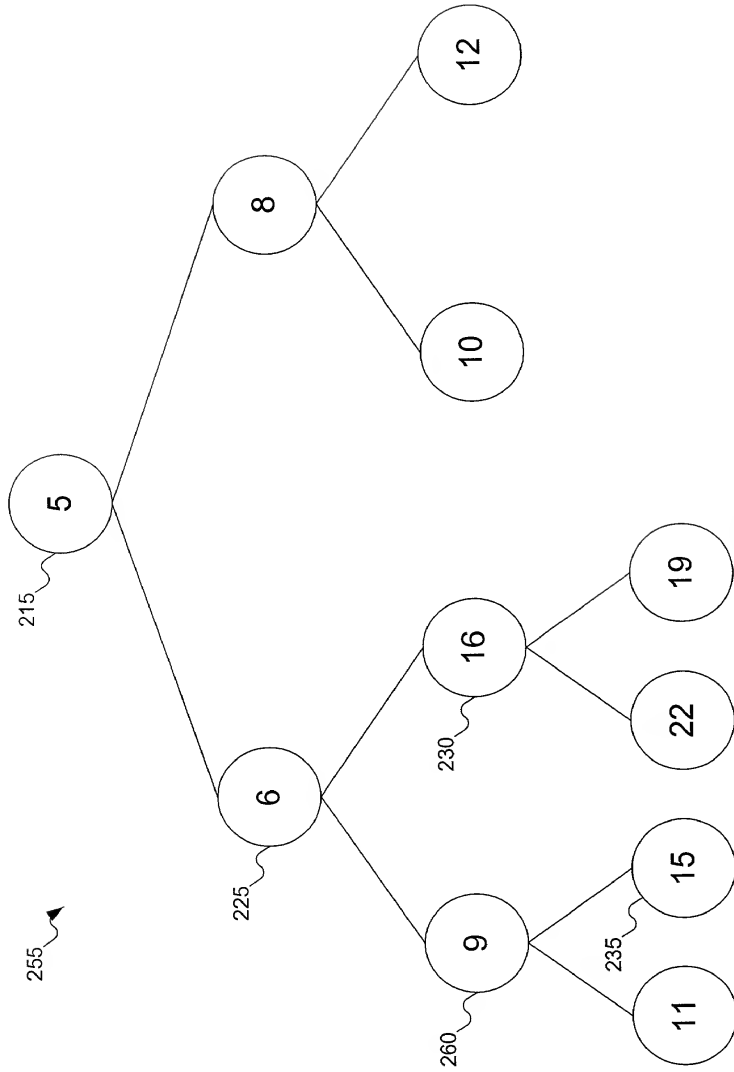


FIG. 2B

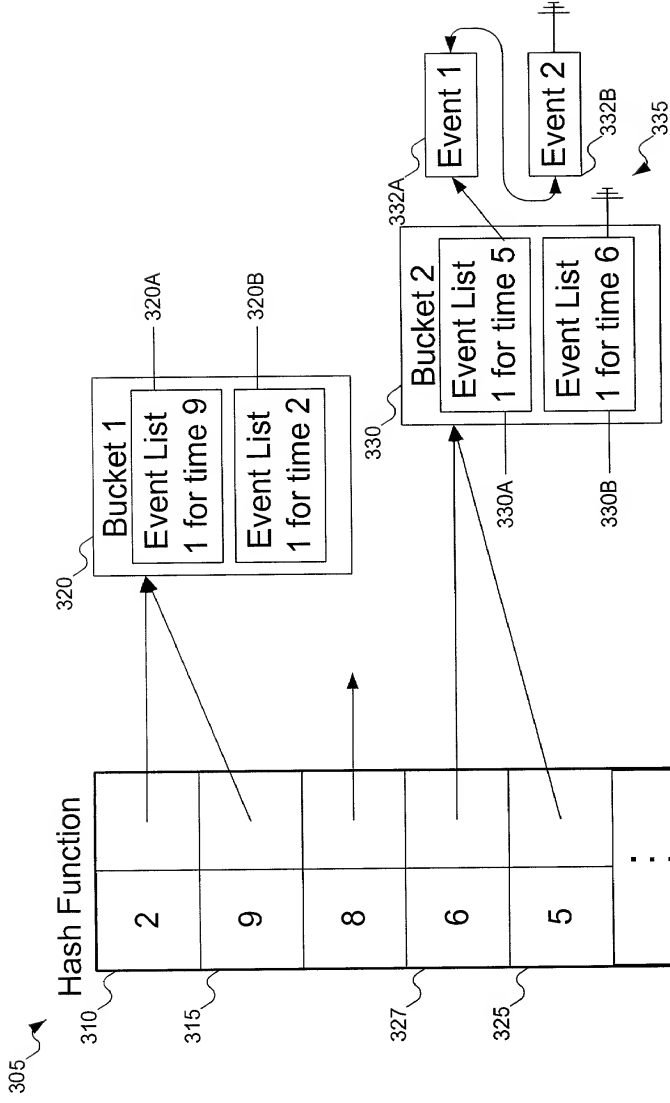


FIG. 3A

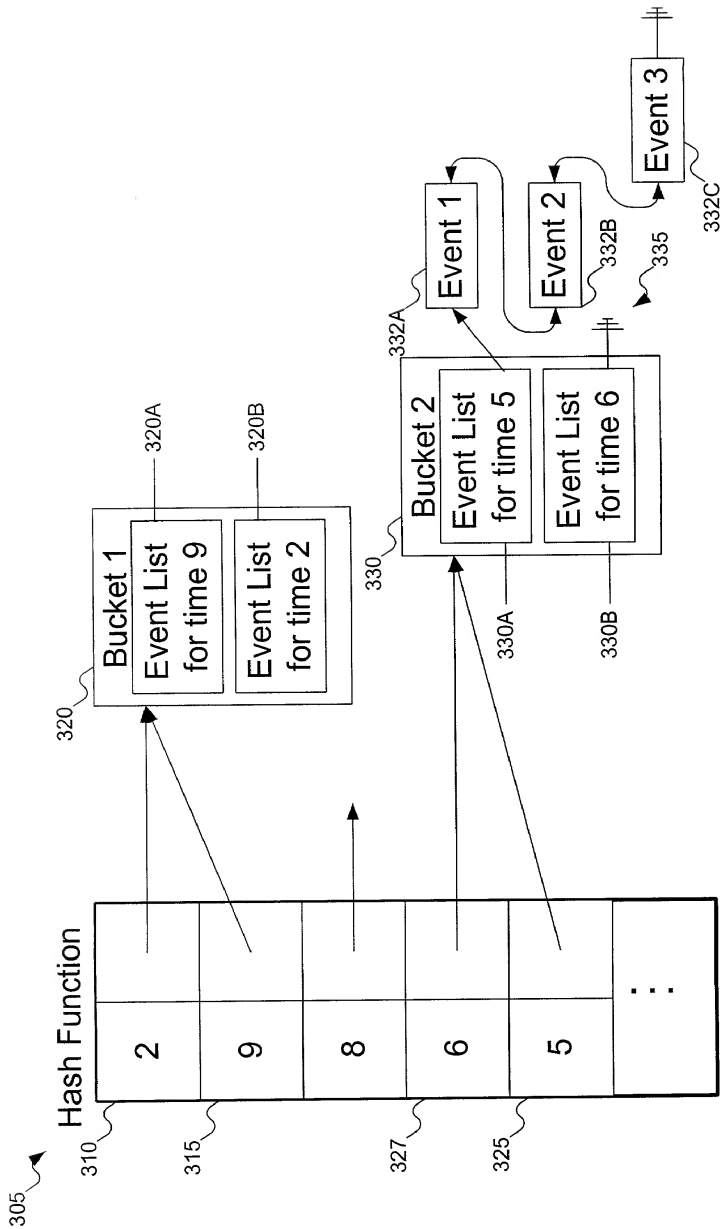


FIG. 3B


```
graph TD
    A{{A}} --> 425{425: Has the end of the timeslot been reached?}
    425 -- No --> B{{B}}
    425 -- Yes --> 430[430: Resume postponed processes]
    430 --> 435{435: Is an analog projection required?}
    435 -- No --> B
    435 -- Yes --> 440[440: Project next analog solution]
    440 --> 445[445: Get the time of the next analog/digital events]
    445 --> B
```

FIG. 4B
(Prior Art)

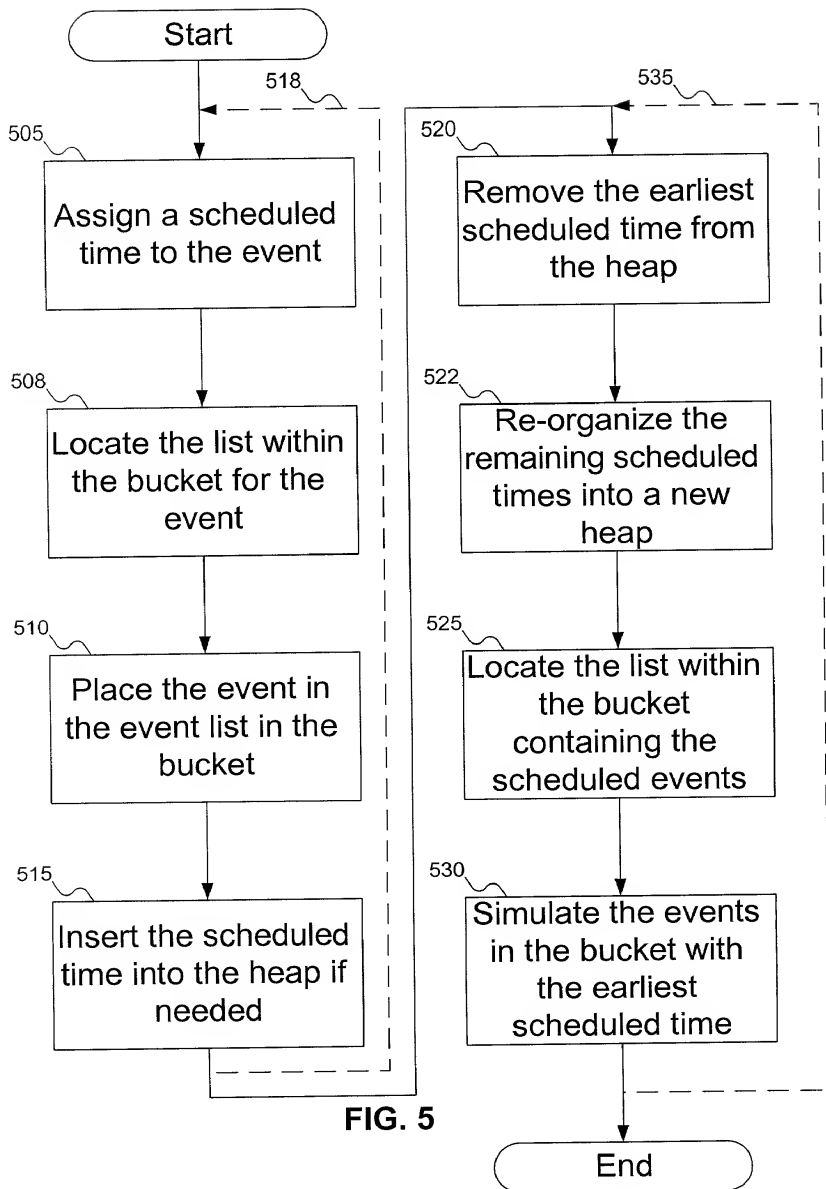


FIG. 5

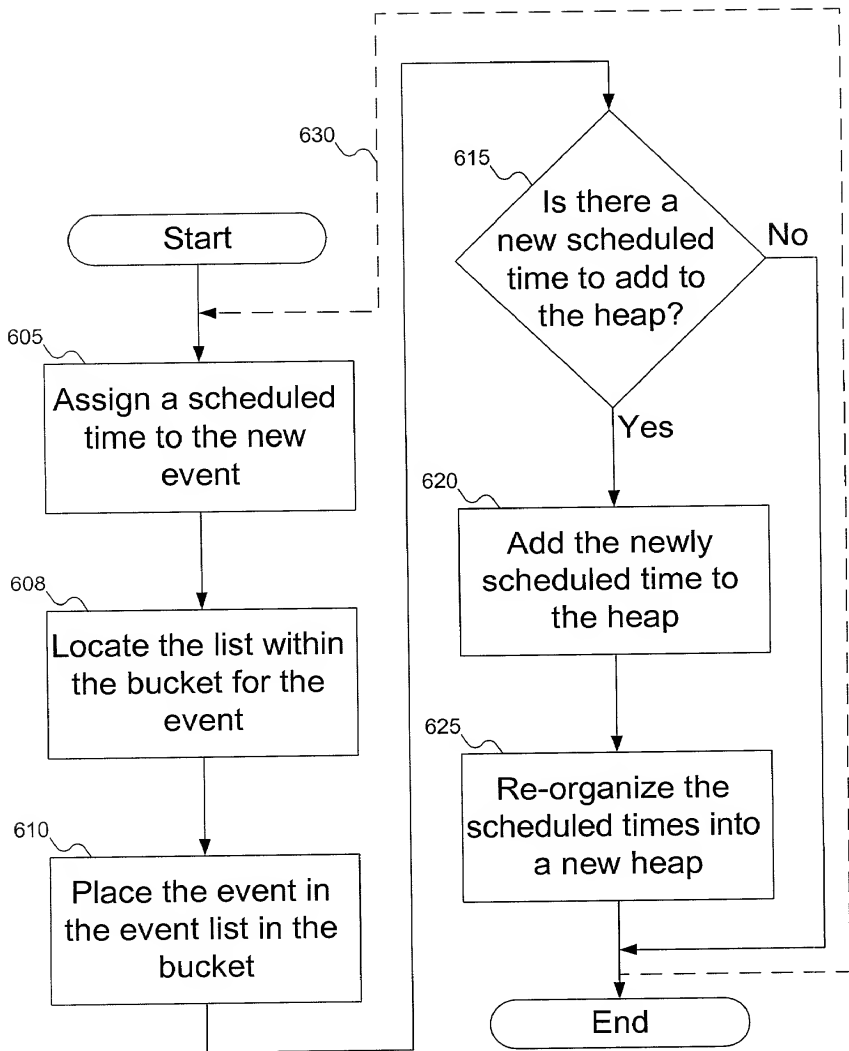


FIG. 6

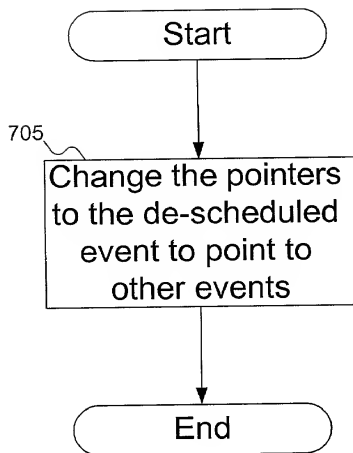


FIG. 7

```
graph TD; Start([Start]) --> 805[Remove a scheduled time from the heap]; 805 --> 808[Re-organize the remaining scheduled times into a new heap]; 808 --> 810[Locate the list within the bucket associated with the scheduled time]; 810 --> 815{Are there events in the event list for the scheduled time?}; 815 -- Yes --> 820[Perform the events in the event list for the scheduled time in the bucket]; 820 --> End([End]); 815 -- No --> 815;
```

The flowchart illustrates the process of event scheduling. It begins with a 'Start' terminal, leading to step 805: 'Remove a scheduled time from the heap'. This is followed by step 808: 'Re-organize the remaining scheduled times into a new heap', and then step 810: 'Locate the list within the bucket associated with the scheduled time'. A decision diamond at step 815 asks 'Are there events in the event list for the scheduled time?'. If the answer is 'Yes', the process proceeds to step 820: 'Perform the events in the event list for the scheduled time in the bucket', which then leads to the 'End' terminal. If the answer is 'No', the process loops back to the decision diamond at step 815.

FIG. 8

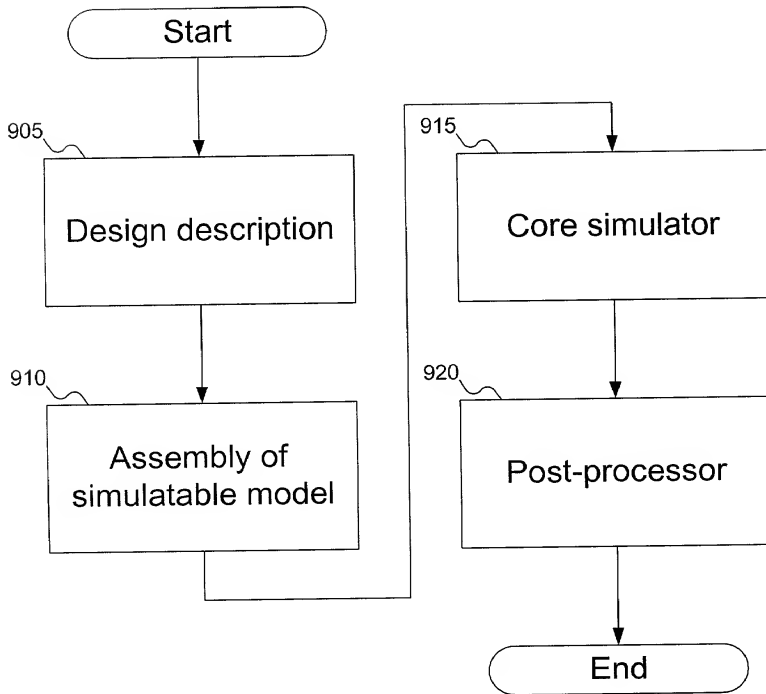


FIG. 9
(Prior Art)

DECLARATION FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled SCHEDULING NON-INTEGRAL SIMULATION TIME FOR MIXED-SIGNAL SIMULATION, the specification of which:

☒ is attached hereto.

☐ was filed on _____ as
Application Serial No. _____

☐ and was amended on _____
(if applicable)

☐ with amendments through _____
(if applicable)

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Sec. 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Sec. 119(a)-(d) of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed: NONE

Prior Foreign Application(s)

Priority Claimed

_____	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>
(Number)	(Country)	(Day/Month/Year Filed)	Yes	No

I hereby claim the benefit under Title 35, United States Code, Sec. 119(e) of any United States provisional application listed below:

Provisional Application No.

Filing Date

60/140,261

June 18, 1999

I hereby claim the benefit under Title 35, United States Code, Sec. 120 of any United States application(s), or Sec. 365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of Title 35, United States Code, Sec. 112. I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Sec. 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application: NONE

(App. Serial No.)

(Filing Date)

(Status -patented, pending, etc.)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or first inventor: Steven S. Greenberg

Inventor's signature:

 3/17/2000
(Date)

Residence:

Beaverton, Oregon

Citizenship:

U.S.A.

Post Office address:

13565 S.W. Hargis Road
Beaverton, Oregon 97008-7432

PATENT APPLICATION
Attorney's Do. No. 1467-13

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Steven S. Greenberg

Serial No.

Filed:

For: SCHEDULING NON-INTEGRAL SIMULATION TIME FOR MIXED-SIGNAL
SIMULATION

BOX PATENT APPLICATION
Assistant Commissioner for Patents
Washington, D.C. 20231

POWER OF ATTORNEY BY ASSIGNEE OF ENTIRE INTEREST
AND REVOCATION OF PRIOR POWERS

I, Howard Ko, Vice President of Engineering, of ANALOGY, INC., having a place of business at 9205 S.W. Gemini Drive, Beaverton, Oregon 97008, assignee of the entire right, title and interest of the above-described U.S. patent application, by the assignment submitted under separate cover for recordal (copy enclosed), represent that I am empowered to sign on behalf of assignee.

As assignee of the above identified application, all powers of attorney previously given are hereby revoked and the following attorneys and/or patent agents are hereby appointed to prosecute and transact all business in the Patent and Trademark Office connected therewith:

Customer No. 20575

Attorney NameRegistration No.

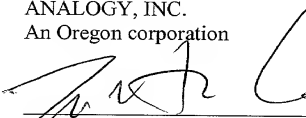
Jerome S. Marger	26,480
Alexander C. Johnson, Jr.	29,396
Alan T. McCollom	28,881
James G. Stewart	32,496
Glenn C. Brown	34,555
Stephen S. Ford	35,139
Gregory T. Kavounas	37,862
Scott A. Schaffer	38,610
Joseph S. Makuch	39,286
James E. Harris	40,013
Graciela G. Cowger	42,444
Ariel Rogson	43,054
Craig R. Rogers	43,888

Direct all telephone calls to Alexander C. Johnson, Jr. at (503) 222-3613 and send all correspondence to:

Marger Johnson & McCollom, P.C.
1030 S.W. Morrison Street
Portland, Oregon 97205

ANALOGY, INC.
An Oregon corporation

Date: 3/17/00



Howard Ko
Vice President of Engineering